**Phase 1: Detailed Requirement Analysis**

**Group Members:** Aimable, Javon, Evan, Femi, Toni

**1. Understanding Requirements**

**Purpose of the Socket Program**

The purpose of our socket program is to create a peer-to-peer (N-way) chat application that allows multiple clients to communicate with each other in real-time through a central server. When a client sends a message to the server, the server will relay the message to other connected clients in real-time except for the initiating sender.

**Chosen Model: Peer-to-Peer (N-way)**

We have chosen to implement the peer-to-peer model for our chat program. This model will offer the following functionality:

* Multiple clients can connect to a central server
* Clients can send messages to all other connected clients
* The server relays messages to all clients except the sender
* A list of connected clients is displayed to each user

**Features of the P2P Model**

1. Multi-client support: The server can handle multiple client connections simultaneously
2. Real-time message relay: Messages are immediately broadcast to all connected clients
3. Client list display: Each client can see a list of other connected clients
4. Server-mediated communication: All messages pass through the server for relay
5. Private messaging: Ability to send messages to specific clients
6. User authentication: Basic username/password system (optional)

**Programming Language: Java**

We have selected Java as our programming language for this project. The reasons for this choice include:

* Team's familiarity with the language
* Cross-platform compatibility
* Strong support for network programming through the java.net package
* Built-in support for multi-threading, essential for handling multiple clients

We plan on using Java Swing for creating the Graphical User Interface (GUI).

**Network Protocol: TCP (Transmission Control Protocol)**

We have chosen to use TCP as our network protocol. Reason for this choice:

* Reliable, ordered, and error-checked delivery of messages
* Connection-oriented protocol, suitable for maintaining persistent connections between clients and the server
* Built-in flow control and congestion control mechanisms

**Members Responsibilities**

**Aimable**

1. Develop core server functionality
2. Coordinate overall project and ensure integration of components
3. Manage GitHub repository
4. Oversee task assignments and ensure equal contribution from all members

**Evan**

1. Implement TCP socket programming for client-server communication
2. Develop protocol for message exchange
3. Map and explain the socket API functions used in the project
4. Assist with integrating network functionality into client and server

**Javon**

1. Design and implement client-side GUI using Java Swing
2. Collaborate with Femi for GUI consistency between client and server
3. Assist in documenting the socket implementation type and function

**Femi**

1. Design and implement server-side GUI using Java Swing
2. Collaborate with Javon for GUI consistency between server and client
3. Create user manual with screenshots for both clients and server

**Toni**

1. Lead testing efforts, focusing on functionality
2. Prepare and execute Wireshark analysis
   * Develop communication scenario for testing
   * Capture packets from both client and server machines
   * Filter packets by IP address
   * Take screenshots of Wireshark displays
   * Provide explanation of which parts show socket communication
3. Implement basic error handling and input validation
4. Compile final project documentation

**Shared Responsibilities**

1. All members participate in code reviews and debugging
2. Each member documents their own code
3. All assist in final integration and testing of the complete system
4. Everyone contributes to the final project report, including:
   * Description of socket implementation
   * GUI screenshots
   * Explanation of the API
   * Wireshark screenshots and analysis